Theoretical Computer Science 411 (2010) 3642–3650

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Parallel-machine scheduling with deteriorating jobs and rejection☆

Shisheng Li*, Jinjiang Yuan

*Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450001, People's Republic of China*

ARTICLE INFO

*Article history:*
Received 21 October 2009
Received in revised form 29 May 2010
Accepted 9 June 2010
Communicated by D.-Z. Du

*Keywords:*
Scheduling
Deteriorating jobs
Rejection
Fully polynomial-time approximation scheme

ABSTRACT

We consider several parallel-machine scheduling problems in which the processing time of a job is a (simple) linear increasing function of its starting time and jobs can be rejected by paying penalties. The objective is to minimize the scheduling cost of the accepted jobs plus the total penalty of the rejected jobs. Three variations of the scheduling cost are considered in this paper. The first is the makespan, the second is the total weighted completion time (for simple linear deterioration), and the third is the total completion time. For the former two problems, we propose two fully polynomial-time approximation schemes to solve them when the number of machines is fixed. For the last problem, we present an optimal $O(n^2)$-time dynamic programming algorithm when the deteriorating rates are equal for all jobs.



## 1. Introduction

For most scheduling problems, it is assumed that the processing times of jobs are fixed parameters [3,25]. However, this assumption is not appropriate for the modelling of many modern industrial processes where the processing time of a job may deteriorate while waiting to be processed. Examples can be found in maintenance scheduling, steel production, cleaning assignment, fire fighting, hospital emergency wards, resource allocation, etc, in which any delay in processing a job may result in an increasing effort to accomplish the job. Such situations may also occur when the machines gradually lose efficiency in the course of processing jobs, so that jobs processed later require a longer processing time. The reader is referred to [20,5,23,24] for more practical motivations to model job deterioration in such a manner.

At the same time, classical scheduling problems routinely assume that all the jobs must be scheduled on some machines so as to optimize a particular optimality criterion. In the real world, however, things may be more complicated. For example, due to limited resources or limited available capacity, the scheduler may choose only a subset of these tasks to be scheduled, while perhaps incurring some penalty for the jobs that are not scheduled, i.e., "rejected". Knapsack is a "pure" instance of the problem, where we only care about the subset to "accept".

In this paper, we study several parallel-machine scheduling problems in which the scheduler can choose a subset of all the tasks to be scheduled in a deteriorating processing environment and reject (not schedule) others by paying penalties.

### 1.1. Problem description

Consider a system with an independent job set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, and $m \geq 2$ identical parallel machines $M_1$, $M_2, \ldots, M_m$. Each machine is continuously available for processing at time 0 onwards and can handle at most one job at a time. Each job $J_j$ ($j = 1, 2, \ldots, n$) is either rejected with a rejection penalty $e_j$ having to be paid, or accepted and processed

☆ Research supported by NSFC (10971201), NSFC-RGC (70731160633) and NSFC (10901142).
* Corresponding author. Tel.: +86 13526896697.
*E-mail address:* lss668@tom.com (S. Li).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.06.008

on one of the $m$ identical parallel machines. Pre-emption is not allowed and the job processing times deteriorate linearly as an increasing function of their starting times. Denote by $S$ and $\bar{S} = \mathcal{J} \setminus S$ the set of accepted jobs and the set of rejected jobs, respectively.

The scheduling problem with linear deteriorating jobs and rejection on $m$ identical parallel machines can be stated as follows. All jobs are simultaneously available at time 0. Each job $J_j$ is associated with a normal processing time $a_j$, and a deteriorating rate $b_j$. The actual processing time $p_j$ of job $J_j$ starting at time $t$ is given by $p_j = a_j + b_j t$. Specifically, for each job, we must decide whether to accept that job or reject it. The accepted jobs are to be scheduled on the $m$ parallel machines so as to optimize a particular criterion, and the rejected jobs are to be paid by their rejection penalties. Denote by $C_j$ the completion time of the accepted job $J_j$. We study the problem of minimizing the makespan of the accepted jobs (i.e., $C_{\max}(S) = \max\{C_j : J_j \in S\}$) plus the total penalty of the rejected jobs. Using the three-field notation of Graham et al. [13], the problem is denoted by $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$, where $m$ is a fixed constant.

When all jobs are associated with a common deteriorating rate $b$ (i.e, $b_j = b$ for $j = 1, 2, \ldots, n$), we consider the problem of minimizing the total completion time (i.e., $\sum_S C_j$) of the accepted jobs plus the total penalty of the rejected jobs. The corresponding problem is denoted by $P|p_j = a_j + bt| \sum_S C_j + \sum_{\bar{S}} e_j$, where $m$ is a variable.

We also consider the scheduling problem with simple linear deteriorating jobs and rejection on $m$ identical parallel machines. Each job $J_j$ is associated with a weight $w_j$, a deteriorating rate $b_j$, and a job-independent release date $t_0 > 0$, i.e., $r_j = t_0$ for $j = 1, 2, \ldots, n$. We assume, as in [23,24,6], that the actual processing time of job $J_j$ starting at time $t$ is $p_j = b_j t$. The assumption "$t_0 > 0$" is made here to avoid the trivial case of $t_0 = 0$ (when $t_0 = 0$, the completion time of each job will be 0). Again, we must decide, for each job $J_j$, whether to accept or reject it. The objective is to minimize the total weighted completion time of the accepted jobs plus the total penalty of the rejected jobs. The problem under consideration is $Pm|p_j = b_j t, r_j = t_0| \sum_S w_j C_j + \sum_{\bar{S}} e_j$.

### 1.2. Relevant previous work

There has been much work on scheduling deteriorating jobs without rejection to minimize a certain objective function. The first study on scheduling jobs with starting time dependent processing times was due to Melnikov and Shafransky [22]. Reviews of this area of research were provided by Alidaee and Womer [2], and Cheng et al. [7]. The monograph by Gawiejnowicz [12] includes detailed discussion of time-dependent scheduling from different perspectives and covers the results not mentioned in these reviews. For the simplest variant of problem $1|p_j = a_j + b_j t|C_{\max}$, Gupta and Gupta [14] showed that sequencing the jobs in nondecreasing order of $a_j/b_j$ is optimal. An FPTAS was presented by Kang and Ng [17] for the NP-hard scheduling problem $Pm|p_j = a_j + b_j t|C_{\max}$. Chen [6] showed that $P2|p_j = b_j t, r_j = t_0| \sum C_j$ is NP-hard and an FPTAS was given by Ji and Cheng [16] for $Pm|p_j = b_j t, r_j = t_0| \sum C_j$. An algorithm $A$ is a $\rho$-approximation ($\rho \geq 1$) algorithm for a minimization problem if it produces a solution that is at most $\rho$ times the optimal one ($\rho$ is also referred to as the worst-case ratio). A family of algorithms $\{A_\epsilon : \epsilon > 0\}$ is called a *fully polynomial-time approximation scheme* (FPTAS) if, for each $\epsilon > 0$, the algorithm $A_\epsilon$ is a $(1 + \epsilon)$-approximation algorithm running in polynomial time in the input size and $1/\epsilon$.

The machine scheduling problem with rejection was first considered by Bartal et al. [4]. They studied both the off-line and on-line versions of scheduling problem with rejection on identical parallel machines. The objective is to minimize the sum of the makespan of the accepted jobs and the total penalty of the rejected jobs. Since then, scheduling problems with rejection have received increasing attention (see, e.g., [26,11,10,15,9,28], etc.)

To the best of our knowledge, the only previous work on deteriorating jobs scheduling that includes rejection is [8]. They focused on the problem of scheduling deteriorating jobs with rejection on a single machine. Their aims are to minimize the makespan, the total weighted completion time and the maximum lateness/tardiness of the accepted jobs plus the total penalty of the rejected jobs. They showed that all these problems are NP-hard. For the simple linear deteriorating jobs, pseudo-polynomial-time algorithms and fully polynomial-time approximation schemes are established to solve them; and optimal polynomial-time dynamic programming algorithms are constructed for some special cases of linear deteriorating jobs. In this paper, we focus our attention on the parallel machines.

### 1.3. Organization of the paper

In Section 2, we consider the problem of minimizing the makespan of the accepted jobs plus the total penalty of the rejected jobs. We present a fully polynomial-time approximation scheme for $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$.

In Section 3, we consider the problem of minimizing the total weighted completion time of the accepted jobs plus the total penalty of the rejected jobs. We first give a fully polynomial-time approximation scheme for $Pm|p_j = b_j t, r_j = t_0| \sum_S w_j C_j + \sum_{\bar{S}} e_j$. Then we provide an optimal $O(n^2)$-time dynamic programming algorithm for $P|p_j = a_j + bt| \sum_S C_j + \sum_{\bar{S}} e_j$.

In Section 4, we conclude the paper and suggest some topics for future research.

## 2. Makespan minimization with rejection

Let $0 < \epsilon \leq 1$ be an arbitrary small rational number, and let $m \geq 2$ be an integral value. Throughout this section, we assume that $a_j$, $b_j$ and $e_j$ ($j = 1, 2, \ldots, n$) are all integral.

When rejection is not allowed, Gupta and Gupta [14] showed that the nondecreasing order of $a_j/b_j$ is optimal for problem $1|p_j = a_j + b_j t|C_{\max}$, which leads to the following lemma.

**Lemma 2.1.** *There exists an optimal job sequence for $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$ such that on each machine the accepted jobs are sequenced in nondecreasing order of $a_j/b_j$.*

Based on Lemma 2.1, it is natural to consider the jobs in nondecreasing order of $a_j/b_j$. So we index the jobs such that $a_1/b_1 \le a_2/b_2 \le \cdots \le a_n/b_n$. We introduce variables $x_j, j = 1, 2, \ldots, n$, where $x_j = k$ if job $J_j$ is scheduled on machine $M_k$, $k = 1, 2, \ldots, m$, and $x_j = 0$ if job $J_j$ is rejected. Let $X$ be the set of all vectors $x = (x_1, x_2, \ldots, x_n)$ with $x_j \in \{0, 1, \ldots, m\}$, and $j = 1, 2, \ldots, n$. We define the following initial and recursive functions on $X$:

$$
\begin{aligned}
&f_0^i(x) = 0, \quad i = 1, 2, \ldots, m;\\
&g_0(x) = 0;\\
&f_j^i(x) = f_{j-1}^i(x) + a_j + b_j f_{j-1}^i(x) \quad \text{for } x_j = i;\\
&f_j^i(x) = f_{j-1}^i(x) \quad \text{for } x_j \neq i;\\
&g_j(x) = g_{j-1}(x) + e_j \quad \text{for } x_j = 0;\\
&g_j(x) = g_{j-1}(x) \quad \text{for } x_j \neq 0;\\
&F(x) = g_n(x) + \max_{i=1,2,\ldots,m} f_n^i(x).
\end{aligned}
$$

Notice that none of $f_j^i$ $(i = 1, 2, \ldots, m)$ and $g_j$ depend on $x_{j+1}, \ldots, x_n$, where $f_j^i(x)$ is the maximum completion time of machine $M_i$ for the accepted jobs among $J_1, J_2, \ldots, J_j$, and $g_j(x)$ is the total penalty of the rejected jobs among $J_1, J_2, \ldots, J_j$. Thus, problem $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$ reduces to the following problem:

Minimize $F(x)$ for $x \in X$.

First, we present the procedure *Partition*$(A, h, \delta)$ proposed by Kovalyov and Kubiak [18,19], where $A \subseteq X$, $h$ is a non-negative integer function on $X$, and $0 < \delta \le 1$. This procedure partitions $A$ into disjoint subsets $A_1^h, A_2^h, \ldots, A_{k_h}^h$ such that $|h(x) - h(x')| \le \delta \min\{h(x), h(x')\}$ for any $x, x'$ from the same subset $A_j^h$, $j = 1, 2, \ldots, k_h$. The following description gives the details of *Partition*$(A, h, \delta)$.

**Procedure** *Partition*$(A, h, \delta)$

Arrange vectors $x \in A$ in order $x^{(1)}, x^{(2)}, \ldots, x^{(|A|)}$ such that $0 \le h(x^{(1)}) \le h(x^{(2)}) \le \cdots \le h(x^{(|A|)})$.

Assign vectors $x^{(1)}, x^{(2)}, \ldots, x^{(i_1)}$ to set $A_1^h$ until a certain $i_1$ is found such that $h(x^{(i_1)}) \le (1+\delta)h(x^{(1)})$ and $h(x^{(i_1+1)}) > (1+\delta)h(x^{(1)})$. If such an $i_1$ does not exist, then set $A_1^h = A$ and stop.

Assign vectors $x^{(i_1+1)}, x^{(i_1+2)}, \ldots, x^{(i_2)}$ to set $A_2^h$ until a certain $i_2$ is found such that $h(x^{(i_2)}) \le (1+\delta)h(x^{(i_1+1)})$ and $h(x^{(i_2+1)}) > (1+\delta)h(x^{(i_1+1)})$. If such an $i_2$ does not exist, then set $A_2^h = A \setminus A_1^h$ and stop.

Continue the above construction until $x^{(|A|)}$ is included in $A_{k_h}^h$ for some $k_h$.

Clearly, Procedure *Partition* requires $O(|A| \log |A|)$ operations to arrange the vectors of $A$ in nondecreasing order of $h(x)$ and $O(|A|)$ operations to provide a partition. The main properties of *Partition*$(A, h, \delta)$ that will be used in development of our FPTAS were presented in [18,19] as follows.

**Lemma 2.2.** *$|h(x) - h(x')| \le \delta \min\{h(x), h(x')\}$ for any $x, x' \in A_j^h$, $j = 1, 2, \ldots, k_h$.*

**Lemma 2.3.** *$k_h \le \log h(x^{|A|})/\delta + 2$ if $h(x^{|A|}) \ge 1$.*

Furthermore, we need the following lemma which was presented in [10].

**Lemma 2.4.** *For any $0 < x \le 1$ and for any integer $n \ge 1$, $(1 + \frac{x}{n})^n \le 1 + 2x$ holds.*

Motivated by the idea of Kovalyov and Kubiak [18,19], a formal description of the FPTAS $\mathcal{A}_\epsilon^m$ for problem $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$ is given below.

**Algorithm** $\mathcal{A}_\epsilon^m$

*Step* 1. (Initialization.) Number the jobs such that $a_1/b_1 \le a_2/b_2 \le \cdots \le a_n/b_n$. Set $Y_0 = \{(0, 0, \ldots, 0)\}$, $g_0 = 0$, $f_0^i = 0$, $i = 1, 2, \ldots, m$, and $j = 1$.

*Step* 2. (Generation of $Y_1, Y_2, \ldots, Y_n$.) Whence $Y_{j-1}$ is determined, we generate set $Y_j'$ by adding $k$, $k = 0, 1, \ldots, m$, in position $j$ of each vector from $Y_{j-1}$. Calculate the following for any $x \in Y_j'$:

$f_j^i(x) = f_{j-1}^i(x) + a_j + b_j f_{j-1}^i(x)$ for $x_j = i$;
$f_j^i(x) = f_{j-1}^i(x)$ for $x_j \neq i$;
$g_j(x) = g_{j-1}(x) + e_j$ for $x_j = 0$;
$g_j(x) = g_{j-1}(x)$ for $x_j \neq 0$.
If $j = n$, then set $Y_n = Y_n'$ and go to Step 3.
If $j < n$, then set $\delta = \epsilon/(2(n+1))$ and perform the following computations.

Call *Partition*$(Y_j', f_j^i, \delta)$ $(i = 1, 2, \ldots, m)$ to partition set $Y_j'$ into disjoint subsets $Y_1^{f^i}, Y_2^{f^i}, \ldots, Y_{k_{f^i}}^{f^i}$.

Call *Partition*$(Y_j', g_j, \delta)$ to partition set $Y_j'$ into disjoint subsets $Y_1^g, Y_2^g, \ldots, Y_{k_g}^g$.

Divide set $Y_j'$ into disjoint subsets $Y_{c_1,\ldots,c_m,d} = Y_{c_1}^{f^1} \bigcap \cdots \bigcap Y_{c_m}^{f^m} \bigcap Y_d^g$, $c_1 = 1, 2, \ldots, k_{f^1}; \ldots; c_m = 1, 2, \ldots, k_{f^m}; d = 1, 2, \ldots, k_g$. In each nonempty subset $Y_{c_1,\ldots,c_m,d}$, choose a vector $x^{(c_1,\ldots,c_m,d)}$ such that

$$g_j(x^{(c_1,\ldots,c_m,d)}) = \min\{g_j(x) : x \in Y_{c_1,\ldots,c_m,d}\}.$$

Set $Y_j := \{x^{(c_1,\ldots,c_m,d)} : Y_{c_1}^{f^1} \bigcap \cdots \bigcap Y_{c_m}^{f^m} \bigcap Y_d^g \neq \emptyset$, where $c_1 = 1, 2, \ldots, k_{f^1}; \ldots; c_m = 1, 2, \ldots, k_{f^m}; d = 1, 2, \ldots, k_g\}$, and set $j := j + 1$. Repeat Step 2.

*Step* 3. (Solution.) Select vector $x^0 \in Y_n$ such that

$$F(x^0) = \min\{F(x) : x \in Y_n\} = \min\{g_n(x) + \max_{i=1,2,\ldots,m} f_n^i(x) : x \in Y_n\}.$$

Let $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ be an optimal solution to problem $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$. Write $x^*[j] = (x_1^*, \ldots, x_j^*, 0, \ldots, 0)$ for $1 \le j \le n$. Define $L = \log \max\{n, 1/\epsilon, a_{\max}, 1 + b_{\max}, e_{\max}\}$, where $a_{\max} = \{a_j : 1 \le j \le n\}$, $b_{\max} = \{b_j : 1 \le j \le n\}$, and $e_{\max} = \{e_j : 1 \le j \le n\}$. We have the following theorem.

**Theorem 2.1.** *Algorithm $\mathcal{A}_\epsilon^m$ finds $x^0 \in X$ for $Pm|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$ such that $F(x^0) \le (1 + \epsilon)F(x^*)$ in $O(n^{2m+2}L^{m+2}/\epsilon^{m+1})$ time.*

**Proof.** Suppose that $x^*[j] = (x_1^*, \ldots, x_j^*, 0, \ldots, 0) \in Y_{c_1,\ldots,c_m,d} \subseteq Y_j'$ for some $j$ and $c_1, \ldots, c_m, d$. By the definition of $\mathcal{A}_\epsilon^m$, such $j$ always exists, for instance $j = 1$. Algorithm $\mathcal{A}_\epsilon^m$ may not choose $x^*[j]$ for further construction; however, for a vector $x^{(c_1,\ldots,c_m,d)}$ is chosen instead of it. By Lemma 2.2, we have

$$|f_j^i(x^*[j]) - f_j^i(x^{(c_1,\ldots,c_m,d)})| \le \delta f_j^i(x^*[j]), \quad i = 1, 2, \ldots, m,$$

and

$$|g_j(x^*[j]) - g_j(x^{(c_1,\ldots,c_m,d)})| \le \delta g_j(x^*[j]).$$

Set $\delta_1 = \delta$. We consider vectors $x^*[j+1] = (x_1^*, \ldots, x_j^*, x_{j+1}^*, 0, \ldots, 0)$ and $\hat{x}^{(c_1,\ldots,c_m,d)} = (x_1^{(c_1,\ldots,c_m,d)}, \ldots, x_j^{(c_1,\ldots,c_m,d)}, x_{j+1}^*, 0, \ldots, 0)$. For $x_{j+1}^* = i$, $i \in \{1, 2, \ldots, m\}$, we have

$$\begin{aligned}
&|f_{j+1}^i(x^*[j+1]) - f_{j+1}^i(\hat{x}^{(c_1,\ldots,c_m,d)})| \\
&= |[f_j^i(x^*[j]) + (a_{j+1} + b_{j+1} f_j^i(x^*[j]))] - [f_j^i(x^{(c_1,\ldots,c_m,d)}) + (a_{j+1} + b_{j+1} f_j^i(x^{(c_1,\ldots,c_m,d)}))]| \\
&= |(1 + b_{j+1})(f_j^i(x^*[j]) - f_j^i(x^{(c_1,\ldots,c_m,d)}))| \\
&\le (1 + b_{j+1})\delta f_j^i(x^*[j]) \\
&\le \delta_1[f_j^i(x^*[j]) + (a_{j+1} + b_{j+1} f_j^i(x^*[j]))] \\
&= \delta_1 f_{j+1}^i(x^*[j+1]).
\end{aligned}$$

For $x_{j+1}^* = 0$, we have

$$\begin{aligned}
|g_{j+1}(x^*[j+1]) - g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)})| &= |[g_j(x^*[j]) + e_{j+1}] - [g_j(x^{(c_1,\ldots,c_m,d)}) + e_{j+1}]| \\
&\le \delta g_j(x^*[j]) \le \delta_1 g_{j+1}(x^*[j+1]).
\end{aligned}$$

Similarly, we have

$$|f_{j+1}^i(x^*[j+1]) - f_{j+1}^i(\hat{x}^{(c_1,\ldots,c_m,d)})| \le \delta_1 f_{j+1}^i(x^*[j+1]) \quad \text{for } x_{j+1}^* \neq i,$$

and

$$|g_{j+1}(x^*[j+1]) - g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)})| \le \delta_1 g_{j+1}(x^*[j+1]) \quad \text{for } x_{j+1}^* \neq 0.$$

As a consequence, we have

$$|f_{j+1}^i(x^*[j+1]) - f_{j+1}^i(\hat{x}^{(c_1,\ldots,c_m,d)})| \le \delta_1 f_{j+1}^i(x^*[j+1]), \quad i = 1, 2, \ldots, m, \tag{1}$$

and

$$|g_{j+1}(x^*[j+1]) - g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)})| \le \delta_1 g_{j+1}(x^*[j+1]). \tag{2}$$

This leads to

$$f_{j+1}^i(\hat{x}^{(c_1,\ldots,c_m,d)}) \le (1 + \delta_1) f_{j+1}^i(x^*[j+1]), \quad i = 1, 2, \ldots, m, \tag{3}$$

and

$$g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) \le (1+\delta_1)g_{j+1}(x^*[j+1]). \tag{4}$$

Assume that $\hat{x}^{(c_1,\ldots,c_m,d)} \in Y_{a_1,\ldots,a_m,b} \subseteq Y'_{j+1}$ and Algorithm $\mathcal{A}^m_\epsilon$ chooses $x^{(a_1,\ldots,a_m,b)} \in Y_{a_1,\ldots,a_m,b}$ instead of $\hat{x}^{(c_1,\ldots,c_m,d)}$ in the $(j+1)$th iteration. From (3) and (4) and by Lemma 2.2, for $i = 1, 2, \ldots, m$, we have

$$|f^i_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) - f^i_{j+1}(x^{(a_1,\ldots,a_m,b)})| \le \delta f^i_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) \le \delta(1+\delta_1)f^i_{j+1}(x^*[j+1]), \tag{5}$$

and

$$|g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) - g_{j+1}(x^{(a_1,\ldots,a_m,b)})| \le \delta g_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) \le \delta(1+\delta_1)g_{j+1}(x^*[j+1]). \tag{6}$$

From (1) and (5), for $i = 1, 2, \ldots, m$, we obtain

$$\begin{aligned}
&|f^i_{j+1}(x^*[j+1]) - f^i_{j+1}(x^{(a_1,\ldots,a_m,b)})| \\
&\le |f^i_{j+1}(x^*[j+1]) - f^i_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)})| + |f^i_{j+1}(\hat{x}^{(c_1,\ldots,c_m,d)}) - f^i_{j+1}(x^{(a_1,\ldots a_m,b)})| \\
&\le (\delta_1 + \delta(1+\delta_1))f^i_{j+1}(x^*[j+1]) \\
&= (\delta + \delta_1(1+\delta))f^i_{j+1}(x^*[j+1]).
\end{aligned} \tag{7}$$

Similarly, from (2) and (6), we obtain

$$|g_{j+1}(x^*[j+1]) - g_{j+1}(x^{(a_1,\ldots,a_m,b)})| \le (\delta + \delta_1(1+\delta))g_{j+1}(x^*[j+1]). \tag{8}$$

Set $\delta_l = \delta + \delta_{l-1}(1+\delta),\ l = 2, 3, \ldots, n-j+1$. From (7) and (8), we have

$$|f^i_{j+1}(x^*[j+1]) - f^i_{j+1}(x^{(a_1,\ldots,a_m,b)})| \le \delta_2 f^i_{j+1}(x^*[j+1]), \quad i = 1, 2, \ldots, m,$$

and

$$|g_{j+1}(x^*[j+1]) - g_{j+1}(x^{(a_1,\ldots,a_m,b)})| \le \delta_2 g_{j+1}(x^*[j+1]).$$

By repeating the above argument for $j+2, \ldots, n$, we can eventually show that there exists $x' \in Y_n$ such that

$$|f^i_n(x') - f^i_n(x^*)| \le \delta_{n-j+1} f^i_n(x^*), \quad i = 1, 2, \ldots, m,$$

and

$$|g_n(x') - g_n(x^*)| \le \delta_{n-j+1} g_n(x^*).$$

From Lemma 2.4, we have

$$\delta_{n-j+1} \le \delta \sum_{j=0}^{n} (1+\delta)^j = (1+\delta)^{n+1} - 1 = \left(1 + \frac{\epsilon}{2(n+1)}\right)^{n+1} - 1 \le \epsilon.$$

Therefore,

$$f^i_n(x') \le (1+\delta_{n-j+1})f^i_n(x^*) \le (1+\epsilon)f^i_n(x^*), \quad i = 1, 2, \ldots, m, \tag{9}$$

and

$$g_n(x') \le (1+\delta_{n-j+1})g_n(x^*) \le (1+\epsilon)g_n(x^*). \tag{10}$$

From (9) and (10), we have

$$\begin{aligned}
F(x') &= g_n(x') + \max_{i=1,2,\ldots,m} f^i_n(x') \\
&\le (1+\epsilon)g_n(x^*) + \max_{i=1,2,\ldots,m} (1+\epsilon)f^i_n(x^*) \\
&= (1+\epsilon)F(x^*).
\end{aligned}$$

So, in Step 3 of Algorithm $\mathcal{A}^m_\epsilon$, vector $x^0$ will be chosen such that

$$F(x^0) \le F(x') \le (1+\epsilon)F(x^*).$$

The time complexity of Algorithm $\mathcal{A}^m_\epsilon$ can be established by noting that the most time-consuming operation of iteration $j$ of Step 2 is a call of procedure *Partition*, which requires $O(|Y'_j| \log |Y'_j|)$ time to complete. To estimate $|Y'_j|$, recall that $|Y'_{j+1}| \le (m+1)|Y_j| \le (m+1)k_{f^1}k_{f^2}\ldots k_{f^m}k_g$. By Lemma 2.3, we have $k_{f^i} \le 2(n+1)\log(na_{\max}(1+b_{\max})^n)/\epsilon + 2 \le 2(n+1)(n+2)L/\epsilon + 2$ for $i = 1, 2, \ldots, m$, and $k_g \le 2(n+1)\log(ne_{\max})/\epsilon + 2 \le 4(n+1)L/\epsilon + 2$. Thus, $|Y'_j| = O(n^{2m+1}L^{m+1}/\epsilon^{m+1})$, and $|Y'_j| \log |Y'_j| = O(n^{2m+1}L^{m+2}/\epsilon^{m+1})$. Therefore, Algorithm $\mathcal{A}^m_\epsilon$ runs in $O(n^{2m+2}L^{m+2}/\epsilon^{m+1})$ time. □

**Corollary 2.1.** *Algorithm $\mathcal{A}^1_\epsilon$ finds $x^0 \in X$ for $1|p_j = a_j + b_j t|C_{\max}(S) + \sum_{\bar{S}} e_j$ such that $F(x^0) \le (1+\epsilon)F(x^*)$ in $O(n^4L^3/\epsilon^2)$ time.*

## 3. Total weighted completion time minimization with rejection

In this section, we study the scheduling problem of minimizing the total weighted completion time of the accepted jobs plus the total penalty of the rejected jobs.

### 3.1. Scheduling simple linear deteriorating jobs

In this subsection, we focus our attention on the simple linear deterioration. We derive a fully polynomial-time approximation scheme for problem $Pm|p_j = b_j t, r_j = t_0|\sum_S w_j C_j + \sum_{\bar{S}} e_j$. This is done by applying the "rounding-the-input-data" technique of Woeginger [27] to a given problem instance. Specifically, the FPTAS is designed by considering the modified deteriorating rates of the accepted jobs. The definition of the modified deteriorating rates involves a geometric rounding technique developed by Afrati et al. [1]. The rounding technique is stated as follows:

For any $\epsilon' > 0$ and $x \geq 1$, if $(1+\epsilon')^{k-1} < x < (1+\epsilon')^k$, then we define $\lceil x \rceil_{\epsilon'} = (1+\epsilon')^k$, $\lfloor x \rfloor_{\epsilon'} = (1+\epsilon')^{k-1}$. If $x$ is an exact power of $(1+\epsilon')$, then $\lfloor x \rfloor_{\epsilon'} = \lceil x \rceil_{\epsilon'} = x$. Note that $\lceil x \rceil_{\epsilon'} \leq (1+\epsilon')x$ for all $x \geq 1$.

For any $0 < \epsilon \leq 1$, we define the modified deteriorating rate of job $J_j$ as $b'_j = \lceil 1 + b_j \rceil_{\epsilon'} - 1$, where $\epsilon' = \epsilon/2n$. Let $\beta_j$ denote the exponent of $1 + b'_j$, i.e., $1 + b'_j = (1+\epsilon')^{\beta_j}$. Then $\beta_j = \frac{\log \lceil 1+b_j \rceil_{\epsilon'}}{\log(1+\epsilon')} = O(\frac{n \log(1+b_j)}{\epsilon})$.

**Lemma 3.1.** *For any $0 < \epsilon \leq 1$, the optimal value for $Pm|p_j = b_j t, r_j = t_0|\sum_S w_j C_j + \sum_{\bar{S}} e_j$ under the modified deteriorating rates is at most $(1+\epsilon)$ times the optimal value for the same problem under the original deteriorating rates.*

**Proof.** Let $\sigma$ be an arbitrary feasible schedule for the accepted jobs in $S$. Consider an arbitrary job $J_j \in S$ scheduled on machine $M_i$, $1 \leq i \leq m$. We denote by $B_j$ the set of jobs that are scheduled before job $J_j$ on machine $M_i$, and by $C'_j$ the completion time of $J_j$ under the modified deteriorating rates in $\sigma$. By the definition of the modified deteriorating rate $b'_j$, we have $1 + b'_j = \lceil 1 + b_j \rceil_{\epsilon'} \leq (1+\epsilon')(1+b_j)$. This yields

$$C'_j = t_0 \prod_{J_k \in B_j \cup \{J_j\}} (1 + b'_k) \leq t_0 (1+\epsilon')^{|B_j|+1} \prod_{J_k \in B_j \cup \{J_j\}} (1 + b_k) \leq (1+\epsilon')^n C_j \leq (1+\epsilon) C_j.$$

The penultimate inequality is justified by noting that $C_j = t_0 \prod_{J_k \in B_j \cup \{J_j\}} (1 + b_k)$ and $|B_j| + 1 \leq |S| \leq n$. The last inequality follows from Lemma 2.4. As a consequence, we have

$$\sum_S w_j C'_j + \sum_{\bar{S}} e_j \leq (1+\epsilon) \sum_S w_j C_j + \sum_{\bar{S}} e_j \leq (1+\epsilon) \left( \sum_S w_j C_j + \sum_{\bar{S}} e_j \right).$$

Since the above inequality is valid for any $S \subseteq \mathcal{J}$ and any feasible schedule $\sigma$, the result holds. □

Mosheiov [23] showed that the nondecreasing order of $b_j/(w_j(1+b_j))$ is optimal for problem $1|p_j = b_j t, r_j = t_0|\sum_{j=1}^n w_j C_j$, which leads to the following lemma.

**Lemma 3.2.** *There exists an optimal job sequence for $Pm|p_j = b_j t, r_j = t_0|\sum_S w_j C_j + \sum_{\bar{S}} e_j$ such that on each machine the accepted jobs are sequenced in nondecreasing order of $b_j/(w_j(1+b_j))$.*

Based on Lemma 3.1, with $(1+\epsilon)$ loss, we can restrict our attention to the scheduling problem $Pm|p_j = b_j t, r_j = t_0|\sum_S w_j C_j + \sum_{\bar{S}} e_j$ under the modified deteriorating rates. Furthermore, by the definition of the modified deteriorating rates, it can be observed that there exists an optimal schedule in which the starting times and completion times of all accepted jobs are in the form of $t_0(1+\epsilon')^l$, where $l$ is a non-negative integer.

Based on Lemma 3.2, it is natural to consider the jobs in nondecreasing order of $\frac{b_j}{w_j(1+b_j)}$. Let us index the jobs such that $\frac{b_1}{w_1(1+b_1)} \leq \frac{b_2}{w_2(1+b_2)} \leq \cdots \leq \frac{b_n}{w_n(1+b_n)}$. For simplicity, define $\Delta_k = (1+\epsilon')^k$. The dynamic programming algorithm is stated as follows:

**Algorithm** $DP_1$

Let $\Phi_j(L_1, L_2, \ldots, L_m)$ denote the optimal value when the jobs in consideration are $J_1, J_2, \ldots, J_j$, and the maximum completion time of jobs on machine $M_k$ is $t_0 \Delta_{L_k}$ for $k = 1, 2, \ldots, m$.

The boundary conditions for the dynamic programming are

$$\Phi_0(L_1, L_2, \ldots, L_m) = \begin{cases} 0, & \text{if } L_1 = L_2 = \cdots = L_m = 0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Now, consider any optimal schedule for jobs $J_1, J_2, \ldots, J_j$, in which the maximum completion time of jobs on machine $M_k$ is $t_0 \Delta_{L_k}$ for $k = 1, 2, \ldots, m$. In any such schedule, there are two possible cases: either job $J_j$ is rejected or job $J_j$ is accepted.

Case 1. Job $J_j$ is rejected. Then $\Phi_j(L_1, L_2, \ldots, L_m) = \Phi_{j-1}(L_1, L_2, \ldots, L_m) + e_j$.

Case 2. Job $J_j$ is accepted. Assume that $J_j$ is scheduled on machine $M_k$, $1 \leq k \leq m$. In this case, for the accepted jobs among $J_1, J_2, \ldots, J_{j-1}$, the maximum completion time of jobs on machine $M_i$ must be $t_0 \Delta_{L_i}$ for $i = 1, \ldots, k-1, k+1, \ldots, m$, and be $t_0 \Delta_{L_k - \beta_j}$ on machine $M_k$. Then $\Phi_j(L_1, L_2, \ldots, L_m) = \Phi_{j-1}(L_1, \ldots, L_{k-1}, L_k - \beta_j, L_{k+1}, \ldots, L_m) + t_0 w_j \Delta_{L_k}$.

Summarizing, we have the following dynamic programming recursion:

$$\Phi_j(L_1, L_2, \ldots, L_m) = \min \begin{cases} \Phi_{j-1}(L_1, L_2, \ldots, L_m) + e_j, \\ \Phi_{j-1}(L_1 - \beta_j, L_2, \ldots, L_m) + t_0 w_j \Delta_{L_1}, \\ \quad \vdots \\ \Phi_{j-1}(L_1, \ldots, L_{k-1}, L_k - \beta_j, L_{k+1}, \ldots, L_m) + t_0 w_j \Delta_{L_k}, \\ \quad \vdots \\ \Phi_{j-1}(L_1, \ldots, L_{m-1}, L_m - \beta_j) + t_0 w_j \Delta_{L_m}. \end{cases}$$

Note that in the above recursive formula, we only need to consider the cases where $L_k \geq 0$ for all $k = 1, 2, \ldots, m$, otherwise no feasible schedule exists.

The optimal objective value is given by $\min\{\Phi_n(L_1, L_2, \ldots, L_m) : 0 \leq L_k \leq \sum_{j=1}^{n} \beta_j,\ 1 \leq k \leq m\}$, and the corresponding optimal schedule can be found by backtracking.

Let $L = \log(1 + b_{\max})$, where $b_{\max} = \max\{b_j : 1 \leq j \leq n\}$. In the dynamic programming recursion, we have $n$ states for $j$, and at most $\sum_{j=1}^{n} \beta_j$ states for each $L_k$, $1 \leq k \leq m$. So the total complexity is $O(n(\sum_{j=1}^{n} \beta_j)^m) = O(n(\sum_{j=1}^{n} n \log(1 + b_j)/\epsilon)^m) = O(n^{2m+1} L^m/\epsilon^m)$. From the above analysis and description, we have the following theorem.

**Theorem 3.1.** *There exists an* FPTAS *for problem* $Pm|p_j = b_j t, r_j = t_0| \sum_S w_j C_j + \sum_{\bar{S}} e_j$, *which runs in* $O(n^{2m+1} L^m/\epsilon^m)$ *time.*

Note that when $w_j = 1$ for $j = 1, 2, \ldots, n$, Ji and Cheng provided an FPTAS for problem $Pm|p_j = b_j t, r_j = t_0| \sum C_j$, which runs in $O(n^{2m+3} T^{m+2}/\epsilon^{m+1})$ time, where $T = \log(\max\{n, 1/\epsilon, 1 + b_{\max}, t_0\})$. The following corollary implies that applying Algorithm $DP_1$ we can obtain an FPTAS for problem $Pm|p_j = b_j t, r_j = t_0| \sum w_j C_j$ with lower running time, even if jobs have distinct weights.

**Corollary 3.1.** *There exists an* FPTAS *for problem* $Pm|p_j = b_j t, r_j = t_0| \sum w_j C_j$, *which runs in* $O(n^{2m-1} L^{m-1}/\epsilon^{m-1})$ *time, where* $L = \log(1 + b_{\max})$.

**Proof.** Define $\Theta_j = \sum_{l=1}^{j} \beta_l,\ j = 1, 2, \ldots, n$. When rejection is not allowed, Lemmas 3.1 and 3.2 still hold. In this case, we do not need to record the state variable $L_m$ for $\Phi_j(L_1, L_2, \ldots, L_m)$ in Algorithm $DP_1$. For job $J_j$, the state variable $L_m$ can be expressed as $L_m = \Theta_j - \sum_{i=1}^{m-1} L_i$. Hence, Algorithm $DP_1$ runs in $O(n(\sum_{j=1}^{n} \beta_j)^{m-1}) = O(n^{2m-1} L^{m-1}/\epsilon^{m-1})$ time. □

Chen [6] proved that no polynomial-time approximation algorithm exists with a constant worst-case ratio for problem $P|p_j = b_j t, r_j = t_0| \sum C_j$, which leads to the following remark.

**Remark.** Unless $P = NP$, there exists no polynomial-time approximation algorithm with a constant worst-case ratio for problem $P|p_j = b_j t, r_j = t_0| \sum_S C_j + \sum_{\bar{S}} e_j$.

### *3.2. Scheduling linear deteriorating jobs*

In this subsection, we consider the scheduling problem of minimizing the total completion time of the accepted jobs plus the total penalty of the rejected jobs, in which the actual processing time of job $J_j$ starting at time $t$ is given by $p_j = a_j + bt$. For the single machine case, this problem has been studied by Cheng and Sun [8]. They propose an $O(n^2)$-time dynamic programming algorithm to solve it. We extend their results to the parallel machines and also obtain an $O(n^2)$-time dynamic programming algorithm. Before presenting our dynamic programming algorithm, we need the following lemma which was obtained by Kuo and Yang [21].

**Lemma 3.3.** *For problem* $1|p_j = a_j + bt| \sum C_j$ *and a job sequence* $\pi = (J_1, J_2, \ldots, J_n)$, *the total completion time of the jobs under schedule* $\pi$ *is*

$$\sum_{j=1}^{n} C_j = \sum_{j=1}^{n} a_j \left( \sum_{i=0}^{n-j} (1 + b)^i \right).$$

For problem $P|p_j = a_j + bt| \sum C_j$, Kuo and Yang [21] showed that there exists an optimal schedule in which jobs are first sorted in nondecreasing order of $a_j$ and then one by one the jobs in the sequence are assigned to each machine in turn. So the following lemma holds.

**Lemma 3.4.** *There exists an optimal schedule for* $P|p_j = a_j + bt| \sum_S C_j + \sum_{\bar{S}} e_j$ *in which the accepted jobs are first sorted in nondecreasing order of* $a_j$ *and then one by one the jobs in the sequence are assigned to each machine in turn.*

**Algorithm** $DP_2$

Let us index the jobs such that $a_1 \leq a_2 \leq \cdots \leq a_n$. Assume the number of machines is $m$. We use $\Phi(j, k)$ to denote the optimal value when the jobs in consideration are $J_j, J_{j+1}, \ldots, J_n$, and exactly $k$ jobs are accepted and scheduled on the $m$ parallel machines.

The boundary conditions for the dynamic programming are

$$\Phi(n,k)=\begin{cases} e_n, & k=0,\\ a_n, & k=1,\\ +\infty, & \text{otherwise.}\end{cases}$$

Now, consider any optimal schedule for jobs $J_{j-1}, J_j, \ldots, J_n$, in which exactly $k$ jobs are accepted and scheduled on the $m$ machines. In any such schedules, there are two possible cases: either job $J_{j-1}$ is rejected or job $J_{j-1}$ is accepted.

Case 1. Job $J_{j-1}$ is rejected. Then $\Phi(j-1,k)=\Phi(j,k)+e_{j-1}$.

Case 2. Job $J_{j-1}$ is scheduled. In this case, by Lemma 3.4, one by one these $k$ accepted jobs in accordance with the accepted sequence are assigned to each machine from behind in turn. So job $J_{j-1}$ must be assigned to the $\lceil k/m\rceil$-position counted from behind on one of the $m$ machines (it can be observed that this machine is uniquely determined by Lemma 3.4), where $\lceil x\rceil$ denotes the smallest integer greater than or equal to $x$. Then, by Lemma 3.3, $\Phi(j-1,k)=\Phi(j,k-1)+a_{j-1}\sum_{i=0}^{\lceil \frac{k}{m}\rceil-1}(1+b)^i$.

Summarizing, we have the following dynamic programming recursion:

$$\Phi(j-1,k)=\min\left\{\Phi(j,k)+e_{j-1},\ \Phi(j,k-1)+a_{j-1}\sum_{i=0}^{\lceil \frac{k}{m}\rceil-1}(1+b)^i\right\}.$$

The optimal value is given by $\min\{\Phi(1,k) : 0\le k\le n\}$, and the corresponding optimal schedule can be found by backtracking.

The dynamic programming recursion has at most $n(n+1)$ states, and each recursion runs in constant time. Hence, the overall time complexity of the algorithm is $O(n^2)$. Consequently, we have the following theorem.

**Theorem 3.2.** *Problem $P|p_j=a_j+bt|\sum_S C_j+\sum_{\bar S}e_j$ can be solved in $O(n^2)$ time.*

## 4. Conclusions

In this paper, we studied several scheduling problems with deteriorating jobs and rejection. The actual processing time of a job is a (simple) linear increasing function of its starting time. When the number of machines is fixed, we presented fully polynomial-time approximation schemes for $Pm|p_j=a_j+b_jt|C_{\max}(S)+\sum_{\bar S}e_j$ and $Pm|p_j=b_jt, r_j=t_0|\sum_S w_jC_j+\sum_{\bar S}e_j$, respectively. Furthermore, we provided an optimal $O(n^2)$-time dynamic programming algorithm for $P|p_j=a_j+bt|\sum_S C_j+\sum_{\bar S}e_j$.

For future research, it would be interesting to focus on scheduling problems with jobs of more general deterioration types. Analysis of the scheduling deteriorating jobs with other objectives is another worthy topic for future research. Furthermore, investigation of the on-line version of this scheduling model is also an interesting research direction.

## References


[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, M. Sviridenko, Approximation schemes for minimizing average weighted completion time with release dates, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, NewYork, 1999, pp. 32–44.

[2] B. Alidaee, N.K. Womer, Scheduling with time dependent processing times: review and extensions, Journal of the Operational Research Society 50 (1999) 711–720.

[3] K.R. Baker, Introduction to Sequencing and Scheduling, Wiley, New York, 1974.

[4] Y. Bartal, S. Leonardi, A.M. Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, in: Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, 1996, pp. 95–103.

[5] S. Browne, U. Yechiali, Scheduling deteriorating jobs on a single processor, Operations Research 38 (1990) 495–498.

[6] Z.L. Chen, Parallel machine scheduling with time dependent processing times, Discrete Applied Mathematics 70 (1996) 81–93.

[7] T.C.E. Cheng, Q. Ding, B.M.T. Lin, A concise survey of scheduling with time-dependent processing times, European Journal of Operational Research 152 (2004) 1–13.

[8] Y.S. Cheng, S.J. Sun, Scheduling linear deteriorating jobs with rejection on a single machine, European Journal of Operational Research 194 (2009) 18–27.

[9] G. Dosa, Y. He, Scheduling with machine cost and rejection, Journal of Combinatorial Optimization 12 (2006) 337–350.

[10] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, J. Wein, Techniques for scheduling with rejection, Journal of Algorithms 49 (2003) 175–191.

[11] L. Epstein, J. Noga, G.J. Woeginger, On-line scheduling of unit time jobs with rejection: minimizing the total completion time, Operations Research Letters 30 (2002) 415–420.

[12] S. Gawiejnowicz, Time-Dependent Scheduling, Springer, Berlin, Heidelberg, 2008.

[13] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 5 (1979) 287–326.

[14] J.N.D. Gupta, S.K. Gupta, Single facility scheduling with nonlinear processing times, Computers and Industrial Engineering 14 (1988) 387–393.

[15] H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection, Mathematical Programming 94 (2003) 361–374.

[16] M. Ji, T.C.E. Cheng, Parallel-machine scheduling with simple linear deterioration to minimize total completion time, European Journal of Operational Research 188 (2008) 342–347.

[17] L.Y. Kang, C.T. Ng, A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs, International Journal of Production Economics 109 (2007) 180–184.

[18] M.Y. Kovalyov, W. Kubiak, A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs, Journal of Heuristics 3 (1998) 287–297.

[19] M.Y. Kovalyov, W. Kubiak, A fully polynomial approximation scheme for the weighted earliness-tardiness problem, Operations Research 47 (1999) 757–761.
[20] A.S. Kunnathur, S.K. Gupta, Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem, European Journal of Operational Research 47 (1990) 56–64.
[21] W.H. Kuo, D.L. Yang, Parallel-machine scheduling with time dependent processing times, Theoretical Computer Science 393 (2008) 204–210.
[22] O.I. Melnikov, Y.M. Shafransky, Parametric problem of scheduling theory, Kibernetika 3 (1979) 55–57 (in Russian).
[23] G. Mosheiov, Scheduling jobs under simple linear deterioration, Computers and Operations Research 21 (1994) 653–659.
[24] G. Mosheiov, Multi-machine scheduling with linear deterioration, INFOR 36 (1998) 205–214.
[25] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice-Hall, Upper Saddle River, NJ, 2002.
[26] S. Seiden, Preemptive multiprocessor scheduling with rejection, Theoretical Computer Science 262 (2001) 437–458.
[27] G.J. Woeginger, When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? Informs Journal on Computing 12 (2000) 57–74.
[28] L.Q. Zhang, L.F. Lu, J.J. Yuan, Single machine scheduling with release dates and rejection, European Journal of Operational Research 198 (2009) 975–978.